
# What does a CPU do before going to work?

## (in an embedded system)

Josef Holzmayr
Embedded Linux Conference Europe - 14.09.2022

# About me

## Josef Holzmayr

Head of Developer Relations at mender.io

Yocto Project Ambassador
OpenEmbedded Social Media Manager
Gitpod.io Community Hero

## Contact Me


josef.holzmayr@northern.tech

@TheYoctoJester

# About you

- Hardware Developers?
- Software Developers?

- Embedded Linux?
- RTOS/bare metal?

# About this presentation


Image attribution: User:-donald- - Wikimedia Commons

# What will NOT be in this presentation

X86.

Boot time optimization.

Code or instructions that you can directly apply to a project.

Advanced boot flows that are specific to specific SoCs, sometimes even single derivatives.

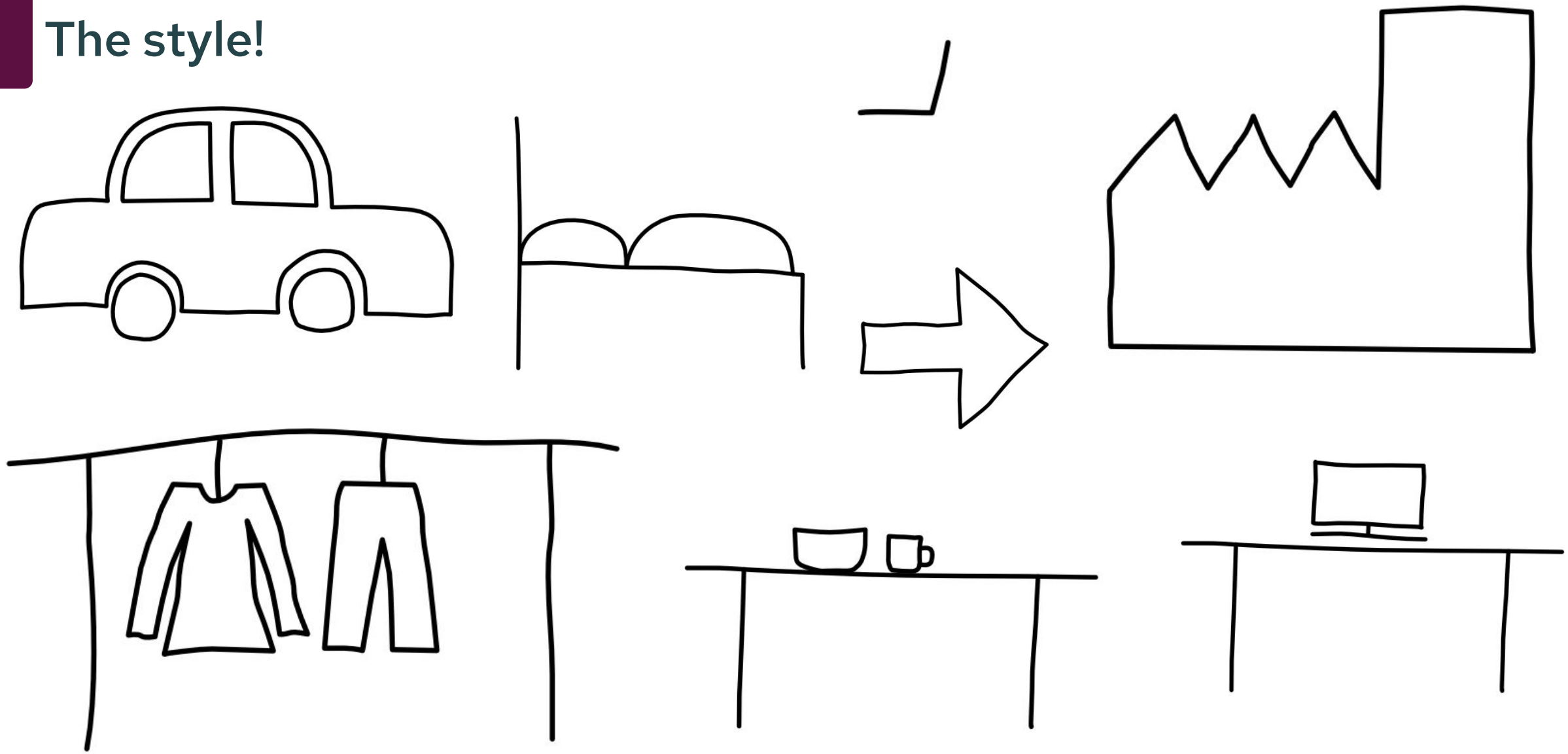
# What will be in this presentation

Low-level technical details, some of the electrical kind.

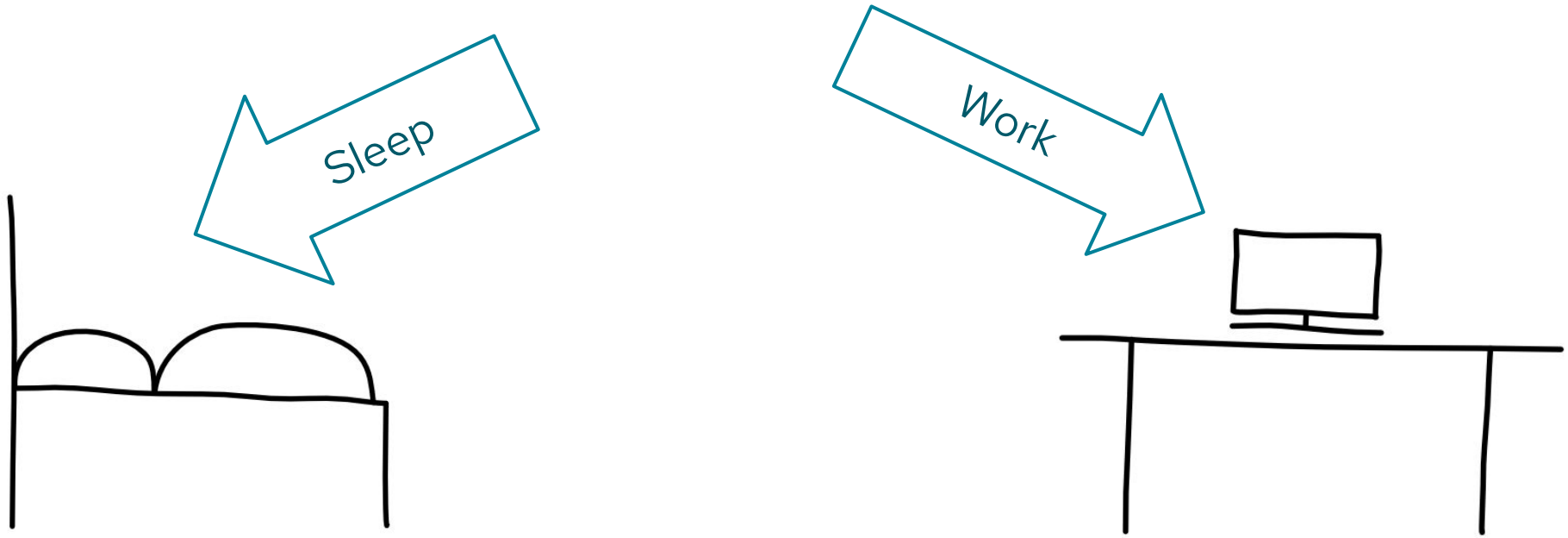
Some historic background.

A metaphor that is hopefully matching well enough so the concepts become clear.

# The style!



# What do you do before going to work?




You wake up in your bed, and then? Which actions to you need to take in order to start working?

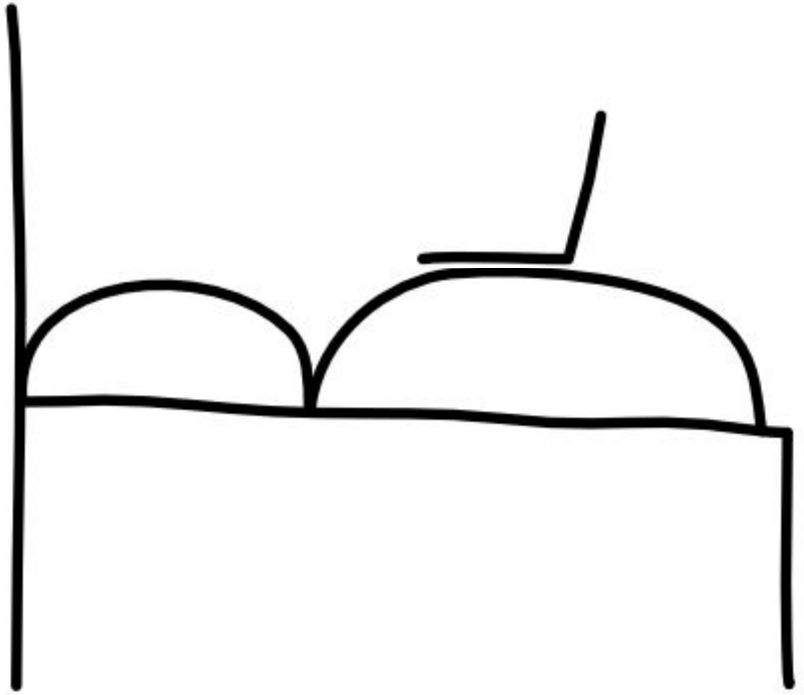
# Stage 1: (Flash-)Microcontrollers - reality

Category:
- Microcontroller (µC) with integrated NOR flash memory and RAM

Booting:
- the CPU runs whatever is at the beginning of the flash memory boot block.
- often some minimal ROM loader that provides In-System-Programming (ISP) and code validity check

Examples:
- primarily ARM Cortex-M0/M0+(/M3/M4) these days
- XTensa (ESP32)
- mostly any proprietary µC with integrated ROM

# Stage 1: (Flash-)Microcontrollers - metaphor

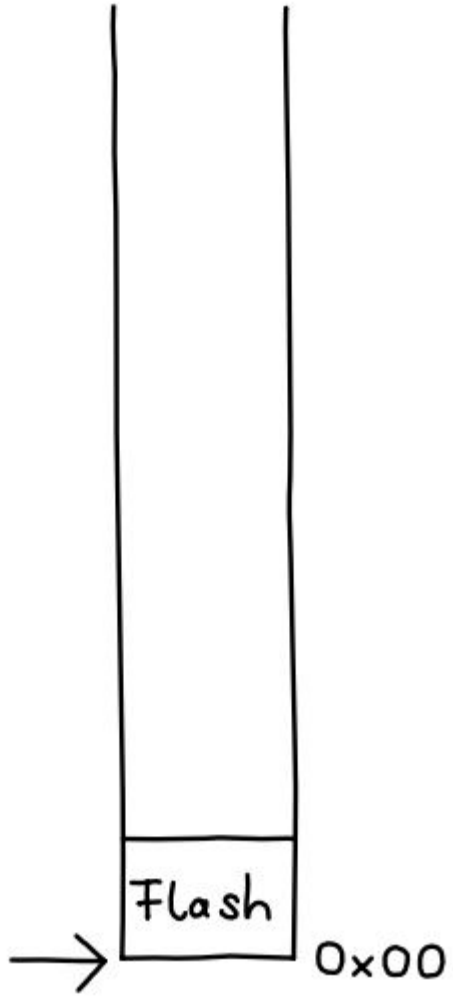


Either work, or sleep. Nothing else. Extreme home office.

# Stage 1: (Flash-)Microcontrollers - technical details

Running:
- "in-place" directly from flash memory: "XIP"
- execution starts at a defined memory location

Initialization:
- (General Purpose) Input/Output pins: "GPIOs"
- Timer(s): the "tick"
- peripherals like buses



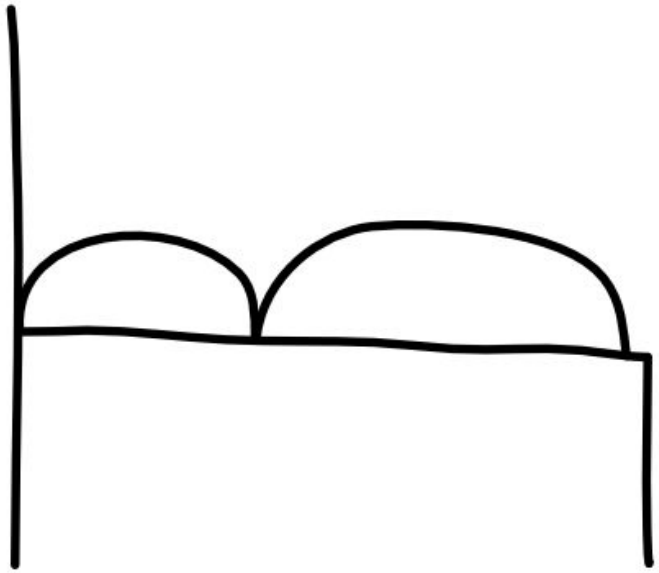

# Stage 2: Microcontrollers with external storage - reality

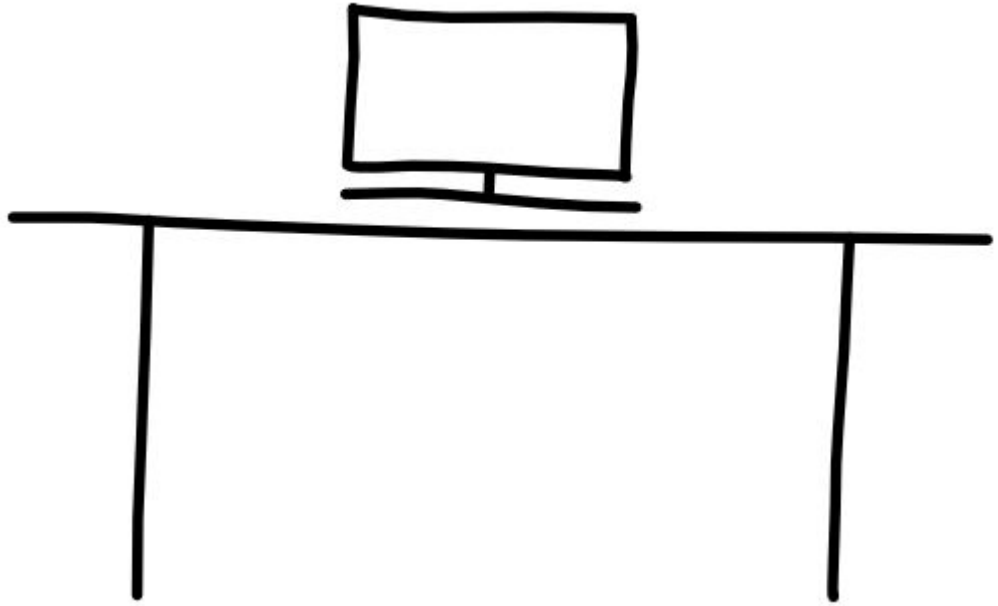
Category:
- Microcontroller (µC) with external (flash) memory

Booting:
- usually a very minimal boot configuration via pin state during reset
- an internal ROM loader acts according to the boot configuration
- might include some device/peripheral initialization: SPI, UART, USB

Examples:
- higher-performance range ARM Cortex-M3/M4
- Kendryte K210 (RISC-V)y

# Stage 2: Microcontrollers with external storage - metaphor





Getting from sleep to work is a short distance, but still it is there. Usually this is configured in actual hardware. Imagine, like a handrail.

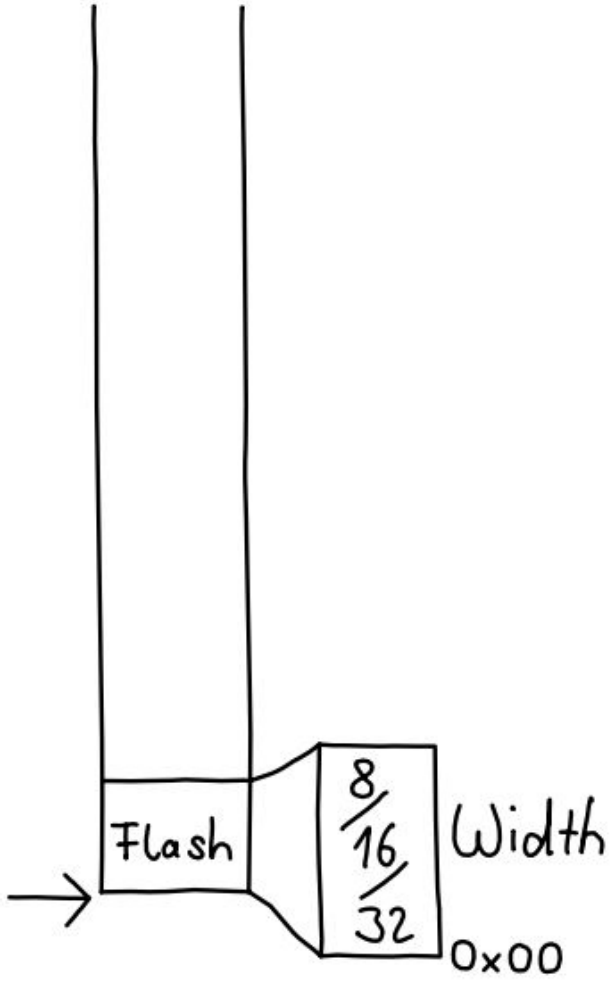
# Stage 2: Microcontrollers with external storage – technical details

Running:

- see Stage 1

Initialization:

- the boot process is usually controlled by pin state, possibly in combination with a fuse-style kind of register. The latter is usually used to disable specific boot paths in order to harden a device. Most common is the selection of the memory interface.



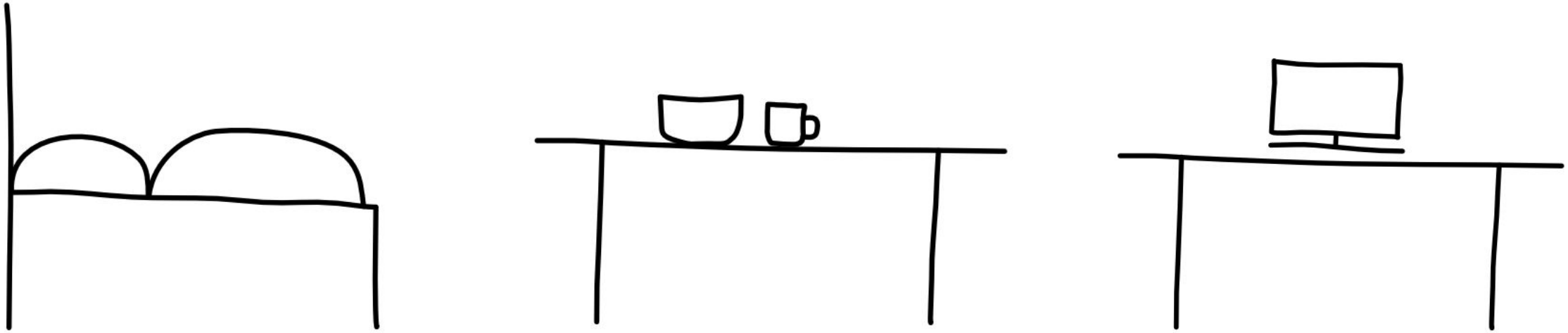
# Stage 3: Microcontrollers using a bootloader – reality

Category:
- Microcontroller (µC) with a software split into loader and application

Booting:
- after the µC-specific startup, the first software part (a custom bootloader) is run
- the bootloader can fulfill different duties, depending on the usecase
  - memory/peripheral initialization
  - program loading/updating mechanisms
  - HW bringup and testing assistance

Examples:
- same as in stage 1 and 2, given a big enough ROM

# Stage 3: Microcontrollers using a bootloader – metaphor

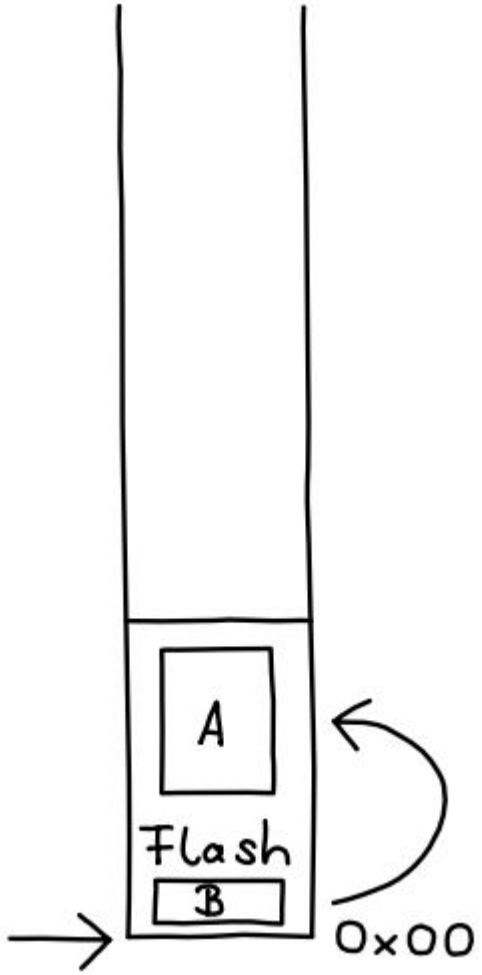


Adding a bootloader brings the first software component into the boot process. This usually means a split into boot/update and application functionality.

# Stage 3: Microcontrollers using a bootloader – technical details

Running:
- bootloader: see stage 1

Initialization:
- the bootloader runs first and does
  - integrity check
  - update trigger check
- the application is usually a free standing binary too, which is just linked to a different section of the ROM storage

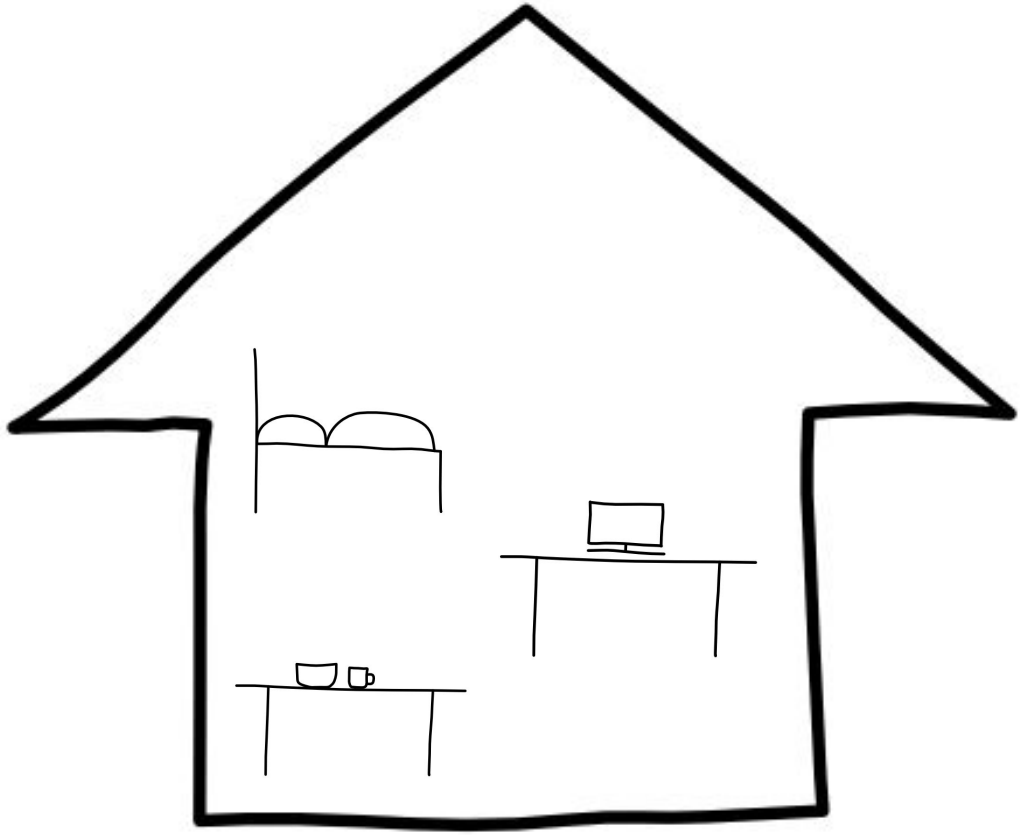


# Intermission: just “bare metal” so far - reality

The term “bare metal” indicates:

- code that runs directly on the CPU
- no management or protection
- no higher libraries
- full control of all peripherals in reach.

# Intermission: just “bare metal” so far - metaphor



“Bare metal” code is like home office. You’re king of the hill, you can do whatever you want.
But your access to advanced equipment is rather limited.

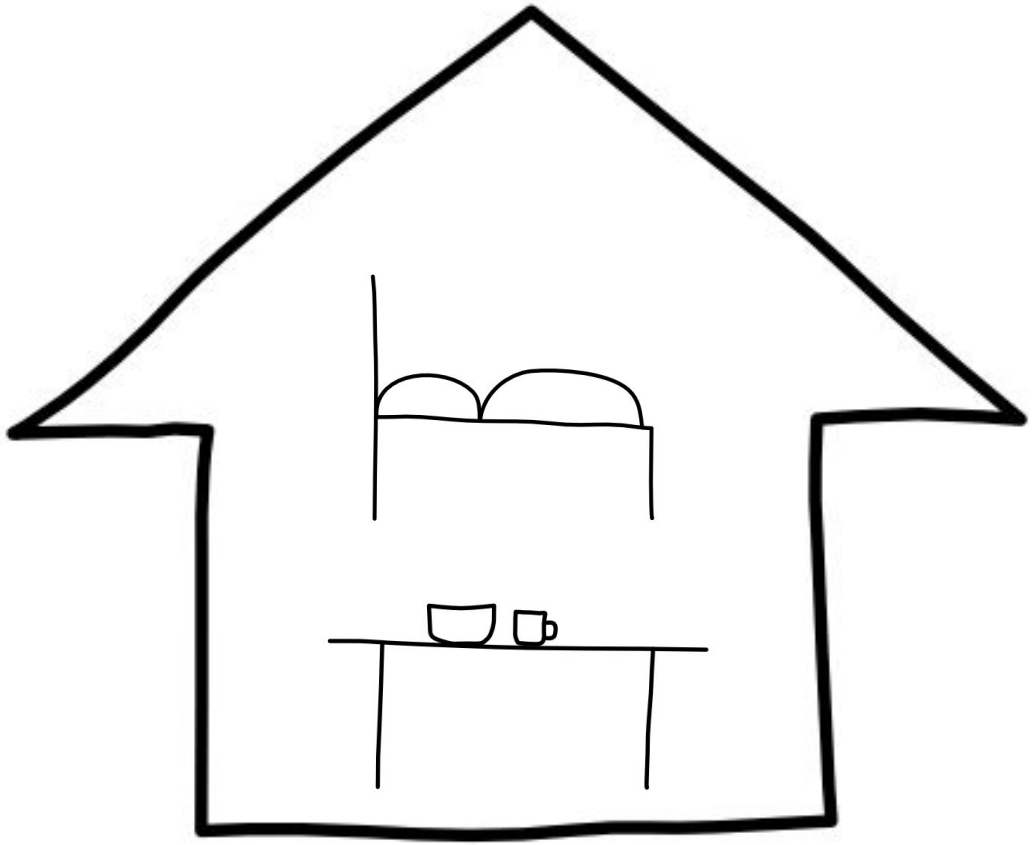
# Stage 4: Microprocessors - reality

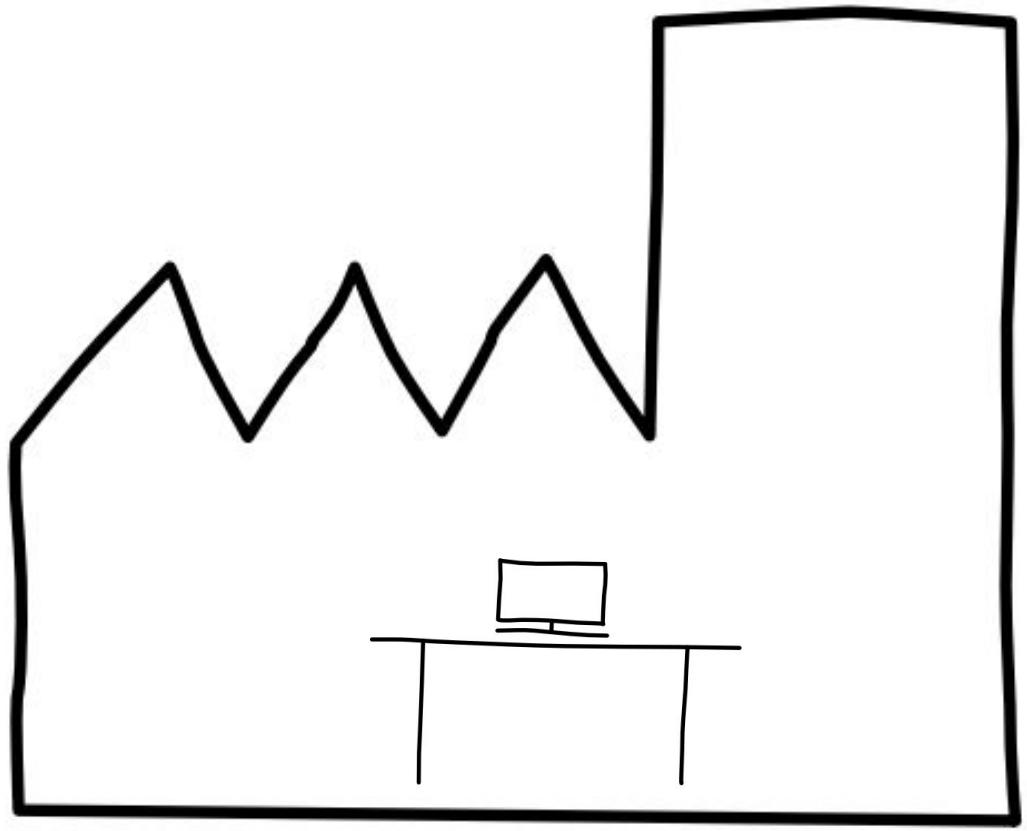
Category:
- Microprocessors (MPUs)
- rule-of-thumb discriminator: having an MMU

Booting:
- up to the bootloader similar to µC
  - hardware configuration of boot process
  - bootloader has similar functionality, but is more commodity (example: U-Boot)
  - the bootloader can be seen as monolithic OS
- bootloader hands off to an operating system (OS), like Linux
  - provides libraries and middleware
  - controls and segregates applications (multiprocessing)

Examples:
- lower end ARM Cortex-A5/A7/A8 MPUs

# Stage 4: Microprocessors – metaphor





After being done with all your morning chores, you head to the company. You're under more control there, but can access professional equipment.

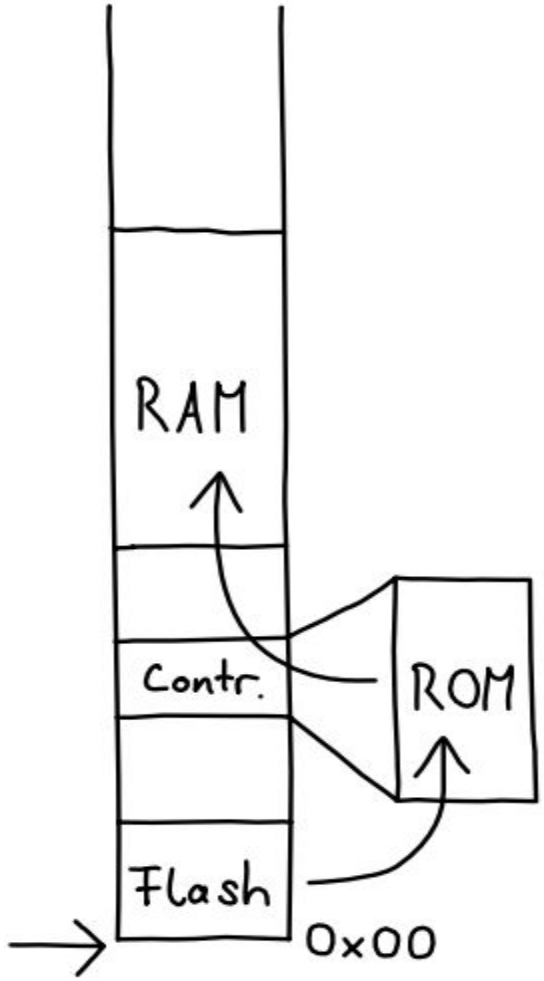
# Stage 4: Microprocessors – technical details

Running:

- bootloader: see stage 1

Initialization:

- the bootloader sets up enough hardware to access more advanced storage and loads the (Linux) application from there into RAM.
- in many cases, the bootloader also loads itself into RAM
- once the application is loaded, it is executed from RAM



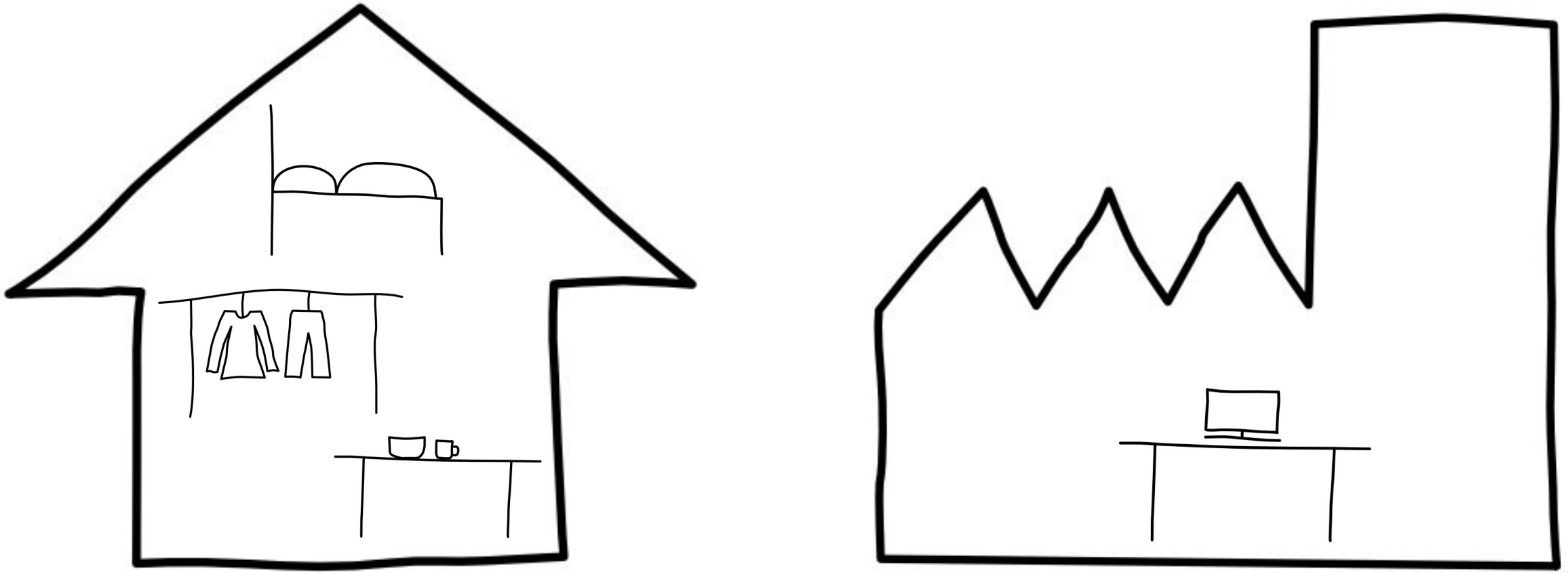
# Stage 5: Advanced microprocessors - reality

Category:
- Microprocessors (MPUs)
- rule-of-thumb discriminator: having an MMU

Booting:
- the bootloader consists of several components that form a chain
  - TPL or SPL: dealing with restricted resources early after reset
  - OpenSBI (RISC-V): system software library loaded during boot
  - Arm Trusted Firmware (ARM): secure software loaded during boot
  - and many, many variations

Examples:
- contemporary ARM and RISC-V processors:
  - Cortex-A8/A9/...
  - SiFive U740

# Stage 5: Advanced microprocessors - metaphor



There's a number of tasks you do before actually leaving home. These sometimes depend on each other, and can be quite complicated.

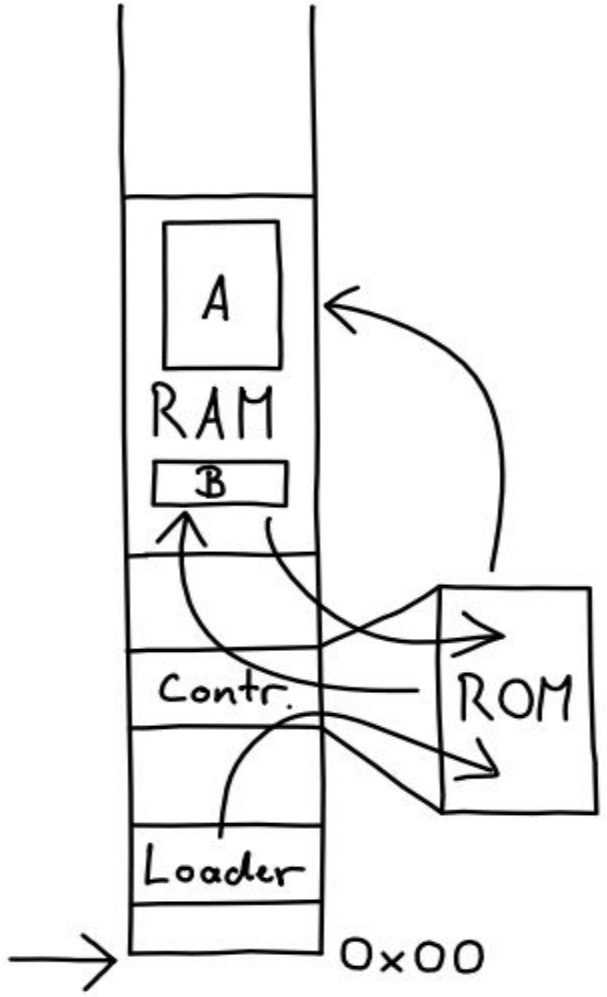
# Stage 5: Advanced microprocessors - technical details

Running:

- the loader which is part of the microprocessor is able to run a bootloader part from specific storage.

Initialization:

- the SPL sets up enough hardware to load the proper bootloader into RAM and execute it
- the proper bootloader proceeds as in stage 4
- once the application is loaded, it is executed from RAM



# Takeaways

Boot sequences form chains that lead up to
- the application on MCUs
- an operating system on MPUs

The more complex the M{C, P}U, the more complex the boot sequence will be.

Early/first stage loading is often highly hardware specific and things become more generic from there onwards.

If there is support for it in hardware, secured software or hypervisors can be loaded during boot before handing over to the operating system.

# Learn more

**Get started now**
docs.mender.io/getting-started

**Join the Mender Hub community**
hub.mender.io

**Mender on Github:**
github.com/mendersoftware/

contact@mender.io

mender.io

@mender_io

company/mender.io


# Thank You

## Q & A